robotron

2

SOFTWARE HANDBUCH

MIKRORECHNER BETRIEBSSYSTEM SCP

Das ideale Nachschlagewerk für den Programmierer

BASIC-Interpreter
BASIC-Compiler

Das vorliegende Nachschlagewerk entspricht dem Stand Juni 1986.

Nachdruck, jegliche Vervielfältigung oder Auszüge daraus sind unzulässig.

Das Softwarehandbuch wurde erarbeitet durch ein Autorenkollektiv der Kammer der Technik im VEB ROBOTRON, Büromaschinenwerk „Ernst Thälmann" Sömmerda .

Dieses Nachschlagewerk wurde mit dem Textprogramm TP erarbeitet und auf dem Mosaikdrucker robotron K6313 ausgedruckt.

Autorenkollektiv: Dr. Ing. Brode, Mathias
Dipl.-Ing. Hubert, Martin
Dipl.-Math. Klein, Udo
Dipl.-Ing.-Ök. Fahr, Klaus

Herausgeber: VEB ROBOTRON Büromaschinenwerk
„Ernst Thälmann" Sömmerda
Weißenseer Straße 52
Sömmerda
DDR - 5230

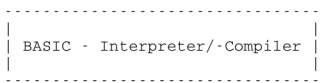
Sömmerda 1986

SOFTWARE - HANDBUCH

für das

Mikrorechner - Betriebssystem SCP

Das ideale Nachschlagewerk für den Programmierer



VEB ROBOTRON Büromaschinenwerk
"Ernst Thälmann" Sömmerda

Vorwort

Das vorliegende Systemhandbuch für Mikrorechner mit dem Betriebssystem SCP ist als ein ideales Nachschlagewerk für Programmierer in Beruf und Hobby geschrieben.

Dieses Handbuch besteht aus den Teilen

- Betriebssystem SCPX

- BASIC-Interpreter/-Compiler

- Assemblerprogrammierung

- Tabellen und Arbeitsblätter

und wird ständig erweitert.

Die Nutzung dieses Nachschlagewerkes setzt Grundkenntnisse über das Betriebssystem SCP sowie der unter Steuerung dieses Systems laufenden Software voraus.

Zur Aneignung von Grundkenntnissen und zur Vertiefung des Wissens über SCP-Software wird vom VEB ROBOTRON, Büromaschinenwerk "Ernst Thälmann", Sömmerda u.a. folgende Dokumentation bereitgestellt.

- Systemhandbuch SCP, Anleitung für den Bediener

- Systemhandbuch SCP, Anleitung für den Programmierer

- Systemhandbuch SCP, Assemblerprogrammierung

- Anwendungsbeschreibung Textprogramm

- Bedienungsanleitung und Sprachbeschreibung BASIC-Interpreter

- Bedienungsanleitung BASIC-Compiler

- Programmieranleitung BASIC

- Bedienungsanleitung und Sprachbeschreibung PASCAL-Compiler

Autorenkollektiv

INHALTSVERZEICHNIS

		Seite
Vorwort		2
Inhaltsverzeichnis		3
1.	Initialisieren BASIC-Interpreter	4
2.	Konstante	4
3.	Variable	5
4.	Zeichensatz	5
5.	Operatoren (in der Reihenfolge der Priorität)	5
5.1.	Arithmetische Operatoren	5
5.2.	Vergleichsoperatoren	6
5.3.	Logische Operatoren	6
5.4.	Zeichenkettenoperatoren	6
6.	Kommandos	6
7.	Programmanweisungen	9
8.	Ein-/Ausgabeanweisungen	12
9.	Anweisungen/Funktionen für Prozessorarbeit	15
10.	Funktionen	16
11.	Tabelle Fehler Codes	20
12.	BASIC-Compiler	21
12.1.	Unterschiede BASIC-Compiler/-Interpreter	21
12.2.	Sprachunterschiede zum Interpreter	21
12.3.	Compilierung von Programmen	22
12.4.	Verbinden von Programmen	23
12.5.	Fehlermitteilungen vom Compiler	23
12.6.	Fehlernachrichten vom Laufzeitsystem	24

1. Initialisieren des BASIC-Interpreters

Eingaben im Betriebssystem SCP (A>) zum Laden und Starten des BASIC-Interpreters:

BASI [<dateiname>] [/F: <anzahl dateien>]
[/M: <höchste Speicherstelle>]
[/S: <max. Satzgröße>]

<dateiname> = Name des vom BASIC-Interpreter zu ladenden und startenden Programms;
<anzahl dateien> = Anzahl der Diskettendateien, die in BASIC gleichzeitig eröffnet werden können (Standard = 3);
<höchste Speicherstelle> = höchste Speicheradresse, die vom BASIC-Interpreter genutzt wird (Standard = gesamter freier Speicher);
<max. Satzgröße> = maximale Satzlänge für Direktzugriffsdateien (Standard = 128).

2. Konstante

- Ganzzahlige Konstante = ganze Zahlen ohne Dezimalpunkt (-32768 ... +32767)
- Gleitkommakonstante = positive oder negative reelle Zahlen (einfache Genauigkeit) mit 7 Stellen Genauigkeit (6 Stellen bei Druck); 10^{-38} ... 10^{+38} z.B. 123.45; -966.; 246.98E-7;; 2.3!
- Gleitkommakonstante = positive oder negative reelle Zahlen (doppelte Genauigkeit) mit 16 Stellen Genauigkeit (16 Stellen bei Druck); 10^{-38} ... 10^{+38} z.B. 1234.6789619; -2.3456789012; 12.4#
- Hexadezimale Konstante = Zahlenwerte im Zahlensystem auf Basis 16 mit Vorsatz "&H" (Zeichenvorrat 0,1,...9,A,B,C,D,E,F) z.B. &H1A4F; &H3C
- Oktale Konstante = Zahlenwerte im Zahlensystem auf Basis 8 mit Vorsatz "&O" oder "&" (Zeichenvorrat 0,1,...7) z.B. &O154; &1777

3. Variable

Namen von Variablen (40 Zeichen signifikant):

Buchstabe [Buchstabe | Ziffer | Punkt...Typkenner]

```
$ String      (0,1,...255 Zeichen)   3+Anzahl Zeichen
% Integer     (-32768...+32767)      2 Byte
! GK (einf.)  (7 Stellen Genauigk.)  4 Byte
# GK (dopp.)  (16 Stellen Genauigk.) 8 Byte
```

4. Zeichensatz

- Alphazeichen = A | B | C |...| Z | a | b | c |...| z
- numerische Zeichen = 0 | 1 | 2 |...| 9
- Sonderzeichen = Ø | = | + | - | * | / | ^ | (|) | %
  # | $ | ! | [|] | , | . | ' | ; | "
  & | ? | < | > | \ | @
- Steuerzeichen = Beginn EDIT-Zustand auf eingegebener Zeile
  ^C Unterbrechung Programmlauf
  ^G Akustisches Signal
  ^H Löschen des zuletzt eingegebenen Zeichens
  ^I Tabellieren (Tab.-Stopp = jede 8. Stelle
  ^O Stopp der Programmausgabe (Start = ^O)
  ^R nochmalige Anzeige der aktuellen Zeile
  ^S Unterbrechung des Programmlaufs
  ^Q Wiederaufnahme der Programmausgabe (nach ^S)
  ^U Streichen der aktuellen Zeile
  <ET> Endetaste
  <ESC> ESC-Taste
  : Trennzeichen für mehrere Anweisungen auf einer Zeile
  . aktuelle Zeile für EDIT, RENUM, DELETE, LIST, LLIST

5. Operatoren (in der Reihenfolge der Priorität)

5.1. Arithmetische Operationen

^	Potenzierung	A^2
-	Negation	-A
*	Multiplikation	A*B
/	Gleitkomma-Division	A/B
\	Ganzzahlige Division	A\B
MOD	Modulo	A MOD B
+	Addition	A+B
-	Subtraktion	A-B

5.2. Vergleichsoperatoren

=	gleich	A=B
<>	ungleich	A<>B
<	kleiner	A<B
>	größer	A>B
<=	kleiner oder gleich	A<=B
>=	größer oder gleich	A>=B

5.3. Logische Operatoren

NOT	Negation	NOT A
AND	logisches AND	A AND B
OR	logisches ODER	A OR B
XOR	logisches Exklusiv-ODER	A XOR B
EQV	logisches Äquivalent	A EQV B
IMP	logische Implikation	A IMP B

5.4. Zeichenkettenoperationen

\+ Verkettung von Zeichenketten (z.B. A$+B$)
Zusätzlich sind alle Vergleichsoperatoren (Pkt. 5.2.) für Zeichenketten zugelassen.

6. Kommandos

AUTO [<zeilennr>[,<schrittweite>]]
Automatische Generierung einer Zeilennummer bei Programmerfassung.
Beispiele: AUTO
AUTO 100, 50

CLEAR [,[<ausdr.1>][,<ausdr.2>]]
Löschen aller Programmvariablen, Schließen aller Dateien und wahlweises Einstellen des Speicherendes und der Stackgröße.
<ausdr.1> = höchste Speicheradresse für BASIC
<ausdr.2> = Einstellen der Stackgröße
Beispiele: CLEAR, 40000
CLEAR, &A000,1000

CONT
Fortsetzung der Programmausführung nach ^C, STOP oder END.

DELETE [<zeilennr>][-[<zeilennr>]]
Löschen von Programmzeilen.
Beispiele: DELETE 100-500
DELETE 100-

EDIT <zeilennummer>
Editieren von Programmzeilen.
Beispiel: EDIT 1100

Edit-Unterbefehle:

[i]Leerzeichen	=	Bewegung des Kursors nach rechts und Anzeige der übergangenen Zeichen (i Stellen).
[i]DEL	=	Bewegung des Kursors nach links und Anzeige der übergangenen Zeichen (i Stellen).
I<text> <ESC>	=	Einfügen von <text> an aktueller Kursorposition (Ende = <ESCAPE>).
X<text> <ESC>	=	Erweitern der Zeile am Zeilenende (analog I).
[i]D	=	Löschung von i Zeichen rechts vom Kursor (einschließlich Kursorposition).
H <ESC>	=	Löschen aller Zeichen rechts vom Kursor und Anfügen von <text> (analog I).
[i]S<ch>	=	Positionieren des Kursors auf das i. Zeichen (<ch>).
[i]K<ch>	=	Positionieren des Kursors auf das i. Zeichen <ch> mit gleichzeitiger Löschung der übergangenen Zeichen.
[i]C<text>	=	Austausch von i Zeichen mit <text> ab aktueller Kursorposition.
<ET>	=	Anzeige der kompletten Zeile, Übernahme der Änderungen und Rücksprung in Kommandoebene.
E	=	Übernahme der Änderungen und Rücksprung in die Kommandoebene.
Q	=	Rücksprung in die BASIC-Kommandoebene ohne Übernahme der Änderungen.
L	=	Anzeige der kompletten Zeile.
A	=	Herstellung des Originalzustandes vor der Editierung.

FILES [<dateiname>]
Anzeige der Dateien auf dem aktuellen bzw. adressierten Laufwerk.
Beispiele: FILES
FILES "B:*.BAS"

KILL <dateiname>
Löschen einer Datei auf Diskette.
Beispiel: KILL"B:ARTIKEL"

LIST [<zeilennr>][-[<zeilennr>]]
Ausgabe des speicherresidenten Programms (oder Teilen des Programms) auf den Bildschirm.
Beispiele: LIST
LIST 200-1000
LIST 120-

LLIST [<zeilennr>][-[<zeilennr>]]
Ausgabe des Programms (oder Teilen des Programms) über den Drucker.
Beispiele: LLIST
LLIST 125-500

LOAD **<dateiname>[,R]**
Laden einer Programmdatei von Diskette in den Speicher.
(R = mit Programmstart)
Beispiele: LOAD "PROG1"
LOAD "B:PROG",R

MERGE **<dateiname>**
Einmischen einer Programmdatei in das speicherresidente Programm.
Beispiele: MERGE "B.UP2"
MERGE "UP1"

NEW
Löschen des speicherresidenten Programms und aller Variablen.

RENUM [[<neue zeilennr>][,[<alte zeilennr>][,<schrittweite>]]]
Neue Durchnumerierung von Programmzeilen im Speicher.
Beispiele: RENUM
RENUM 100,10,50
RENUM 100,,20

RESET
Ermöglicht die Aufzeichnung von Daten auf Kassette nach Diskettenwechsel.

RUN **[<zeilennr>]**
Start des Programms ab <zeilennr>.
Beispiele: RUN
RUN 150

RUN **<dateiname>,[,R]**
Laden einer Programmdatei in den Speicher und Starten des Programms (mit "R" bleiben alle Dateien geöffnet).
Beispiele: RUN "PROG"
RUN "B:PR1",R

SAVE **<dateiname>[,A | P)]**
Ausgabe einer Programmdatei vom Speicher auf Diskette:
"A" = Ausgabe im ASCII-Code
"P" Ausgabe im verschlüsselten Binärformet
Beispiele: SAVE "A:PROG"
SAVE "PR1",A
SAVE "B:PR2",P

SYSTEM
Schließen aller Dateien und Rückkehr zum SCP.

TRON
Einschalten des TRACE-Modus.

TROFF
Ausschalten des TRACE-Modus.

7. Programm-Anweisungen

CHAIN [MERGE]<dateiname>[[,<zeilennr>][,ALL][,DELETE<bereich>]]
Nachladen eines Programms von Diskette und Starten des Programms ab <zeilennr>.
<dateiname> = Name der zu ladenden Programmdatei
<zeilennr> = Zeilennummer, mit der das Programm gestartet wird
MERGE = nachzuladendes Programm überlagert das speicherresidente Programm;
ALL = alle Variablen des laufenden Programms werden an das gerufene Programm übergeben;
DELETE = Löschen von Programmzeilen (<bereich>) vor dem Nachladen.
Beispiele: CHAIN "PR1",1000
CHAIN MERGE "B:DVS",650,ALL,DELETE 800-850

COMMON <variablenliste>
Übergabe von Werten (<variablenliste>) zwischen Programmen bei CHAIN-Anweisungen ohne ALL.
Beispiel: COMMON A,B(),C$

DEF FN<name>[(<parameterliste>)] = <funktionsdefinition>
Definieren von Nutzerfunktionen (Funktionsnamen beginnen immer mit FN).
Beispiel: DEF FNAP(a,b)=SQR(a*a+b*b)

DEFINT <bereich der buchstaben>
DEFSNG <bereich der buchstaben>
DEFDBL <bereich der buchstaben>
DEFSTR <bereich der buchstaben>
Definition von Buchstaben/Buchstabenbereichen für INTEGER (INT), Gleitkomma einfacher (SNG) und doppelter (DBL) Genauigkeit, sowie für Zeichenketten (STR).
Beispiele: DEFINT a
DEFSNG c-k
DEFDBL m,p-r
DEFSTR s-v,z

DIM <liste indizierte variable>
Definition der Maximalwerte für die Indizierung der Feldvariablen.
Beispiel: DIM a$(5), a(15,10)

END
Beenden des Programmlaufes und Schließen aller Dateien.

ERASE <liste feldvariable>
Löschen von Feldern.
Beispiel: ERASE a,b$

ERROR <integerausdruck>
Simulation eines BASIC-Fehlers und Definition von Fehlercodes durch den Nutzer.
Beispiel: ERROR 17

FOR <variable> = x TO y [STEP z]
<anweisungen>
NEXT <variable>
Mehrfaches Wiederholen einer Folge von Anweisungen
(< = Anfangswert, y = Endwert, z = Schrittweite.
Beispiel:
```
FOR i=1 TO 10: PRINT i: NEXT i
FOR I=12 TO 122 STEP2
a=a+i*b
LPRINT a, f(i), i
NEXT i
```

GOSUB <zeilennr>
Verzweigungen zu einem BASIC-Unterprogramm, beginnend mit <zeilennr>
Beispiel: GOSUB 120

GOTO <zeilennr>
Unbedingte Verzweigung zur <zeilennr>
Beispiel: GOTO 155

IF <ausdruck> THEN <anweisung>[ELSE <anweisung>]
In Abhängigkeit vom Ergebnis <ausdruck> wird die Abarbeitung der Anweisungen ausgeführt:
<ausdruck>="wahr": Ausführung der Anweisungen nach THEN;
<ausdruck>="falsch": Ausführung der Anweisungen nach ELSE oder der folgenden Anweisung.

IF <ausdruck> GOTO <zeilennr> [ELSE<anweisung>]
Wenn der <ausdruck> "wahr" ist, wird ein Sprung zu <zeilennr> ausgeführt; ist der <ausdruck> "falsch", werden die <anweisungen> nach ELSE oder die folgende Anweisung ausgeführt.
Beispiele:
```
IF a=10 THEN b=b+c
IF a>b GOTO 120 ELSE a=b
IF (a+6) <= (4-i) THEN 350
```

[LET] <varaible> = <ausdruck>
Einer Variablen wird der Wert eines Ausdruckes zugewiesen (LET ist optional).
Beispiele:
```
a=a+b*c
LET c=d
```

MID$ (<zeichenkettenexp1>,n[,m]) = <zeichenkettenexp2>
Ersetzen eines Teils der Zeichenkette1 durch die Zeichenkette2.
n = Anfangsposition in <zeichenkettenexp1>
m = Anzahl der Zeichen
Beispiel: MID$(a$,4,2)="km"

NULL <exp>
Ausgabe einer Anzahl von null-Codes (&H00) nach einer Zeilenschaltung.

ON ERROR GOTO <zeilennr>
Spezifizierung der Anfangsposition (<zeilennr>) einer Behandlungsroutine.
Beispiel: ON ERROR GOTO 1000

ON <ausdruck> GOTO <liste zeilennr>
Verzweigung zu einer spezifizierten Zeilennummer in Abhängigkeit vom Wert des Ausdrucks.
Beispiel: ON i GOTO 200,400,600,780

ON <ausdruck> GOSUB <liste zeilennr>
Verzweigung in ein Unterprogramm in Abhängigkeit vom Wert des Ausdruckes.
Beispiel: ON i+j GOSUB 120,330,430

OPTION BASE n
Festlegen des kleinsten Wertes für die Indizierung der Feldvariablen (n = 0,1; Standard = 0)

RANDOMIZE [<ausdruck>]
Neueinstellung des Zufallsgenerators. Fehlt der <ausdruck>, dann wird eine Zahleneingabe über die Tastatur erwartet.
Beispiel: RANDOMIZE 12

REM <kommentar>
Anweisung zur Einfügung von Kommentaren in Programme.
Kommentare am Zeilenende können auch durch einen Apostroph (') eingeleitet werden.
Beispiele: REM Listenausdruck
'Das ist ein Kommentar

RESTORE [<zeilennr>]
Positionierung des DATA-Zeigers auf das erste Element der DATA-Anweisung in <zeilennr>. Fehlt <zeilennr>, dann wird die DATA-Anweisung mit der niedrigsten Zeilennummer verwendet.
Beispiel: RESTORE 560

RESUME | RESUME 0 | RESUME NEXT | RESUME <zeilennr>
Fortsetzung des Programms Abarbeitung einer Fehlerroutine bei:
- RESUME (RESUME 0): mit der Anweisung, die den Fehler verursacht hat;
- RESUME NEXT. mit der Anweisung, die der fehlerhaften Anweisung folgt;
- RESUME <zeilennr>: mit der <zeilennr>.

Beispiele: RESUME
RESUME NEXT
RESUME 105

RETURN
Rücksprung aus einem mit GOSUB gerufenen Unterprogramm; Fortsetzung mit der Anweisung nach GOSUB.

STOP
Unterbrechung der Programmabarbeitung

SWAP <variable>,<variable>
Austausch der Werte von zwei Variablen
Beispiel: SWAP a,b

WHILE <ausdruck>
<anweisungen>
WEND
Ausführen der Anweisungen in der WHILE...WEND-Schleife, solange der <ausdruck> einen wahren Wert liefert.
Beispiele:
```
WHILE a>10: PRINT a: a=a+1: WEND
WHILE (i-6) <= k
PRINT i,k
k:=k+2
WEND
```

8. Ein-/Ausgabeanweisungen

CLOSE [[#]<dateinr> [,[#]<dateinr> ...]]
Schließen von Diskettendateien.
Wird kein Argument angegeben, werden alle offenen Dateien geschlossen.
Beispiel:
```
CLOSE
CLOSE #3,4
```

DATA <liste der konstanten>
Angabe der numerischen und Zeichenkettenkonstanten, die durch READ-Anweisungen gelesen werden.
Beispiel: `DATA 2.987, 4, "ABC"`

FIELD [#]<dateinr>,feldbreite> AS <zeichenkettenvariable>...
Speicherplatzreservierung für einen Puffer, in dem Variablen für eine Direktzugriffsdatei abgelegt werden.
Beispiel: `FIELD #1,3 AS a$, 2 AS INTG$, 4 AS GKE$, 8 AS GKD$`

GET [#] -dateinr> [,<satznr>]
Lesen eines Satzes von der Direktzugriffsdatei in den Puffer. Fehlt <satznr>, dann wird der nächste Satz gelesen.
Beispiele:
```
GET #1
GET #1,12
GET #1,snr
```

INPUT [;] [<zeichenkettenkonst> {;|,}] <variablenliste>
Eingabe von Daten über die Tastatur in Variable:
```
<zeichenkettenkonst>        = Bedienernachricht auf Bildschirm
; nach INPUT                = keine Zeilenschaltung nach ET
; nach <zeichenkettenkonst> = Ausgabe Fragezeichen
, nach <zeichenkettenkonst> = Unterdrückung Fragezeichen
```
Beispiele:
```
INPUT "A,B"; a$,b#
INPUT; "Preis: ",preis
```

INPUT #<dateinr>,<variablenliste>
Einlesen von Daten aus einer sequentiellen Datei und Wertzuweisung an Variable.
Beispiel: `INPUT #1,a,b,c`

KILL <dateiname>
Löschen von Diskettendateien
Beispiele:
```
KILL "A:ART.DAT"
txt$ = "C:KUNDEN"
KILL txt$
```

LINE INPUT [;] [<zeichenkettenkonst>{;|,}] <zeichenkettenvar>
Eingabe einer beliebigen Zeichenkette bis 254 Zeichen über die Tastatur. Die <zeichenkettenkonst> wird als Bedienernachricht auf dem Bildschirm ausgegeben.
Beispiele: LINE INPUT a$
LINE INPUT "Name: ";n$

LINE INPUT #<date-nr>, <zeichenkettenvariable>
Einlesen einer ganzen Zeile ohne Trennzeichen aus einer sequentiellen Datei in eine Zeichenkettenvariable.
Beispiel: LINE INPUT #3,BB$

LSET <zeichenkettenvariable> = <zeichenkettenausdruck>
Linksbündiges Eintragen der Daten (<zeichenkettenausdruck>) in den Pufferspeicher für Direktzugriffsdateien.
Beispiele: LSET a$="ANTON"
LSET b$=MKS$(maxw)

OPEN <modus>,[#]<dateinr>,<dateiname>[,<satzlänge>]
Eröffnung einer Diskettendatei.
<modus> = "R": Direktzugriffsdatei (Ein-/Ausgabe)
"I": sequentielle Datei (Eingabe)
"O": sequentielle Datei (Ausgabe)
Als Standardlänge (<satzlänge>) bei Direktzugriffsdateien werden 128 Bytes angenommen.
Beispiele: OPEN "R",#1,"KUNDE",80
OPEN "I",#2,"A:ERF.DAT"

PRINT [<liste der ausdrücke>] [{,|;}]
LPRINT [<liste der ausdrücke>] [{,|;}]
Die Werte der ermittelten Ausdrücke werden auf dem Drucker (LPRINT) oder dem Bildschirm (PRINT) ausgegeben.
Trennzeichen in <liste der ausdrücke>:
, = Nächster Wert wird am Beginn der folgenden TAB-Zone ausgegeben.
; = Nächster Wert wird unmittelbar nach dem letzten Wert ausgegeben.

Ein Semikolon oder Komma am Ende der <liste der ausdrücke> bewirkt, daß nach Ausgabe des letzten Wertes keine Zeilenschaltung ausgeführt wird.
Wenn <liste der ausdrücke> fehlt, dann wird eine Leerzeile ausgegeben.
Anstelle von PRINT kann "?" verwendet werden.
Beispiele: PRINT "Text: ",txt$, "Preis: ",preis$
PRINT a,b+c/3;
? i%; j%+k%
LPRINT "Datum: ",dat$

PRINT USING <formatkette>;<liste der ausdrücke>
LPRINT USING <formatkette>;<liste der ausdrücke>
Ausgabe von Zeichenketten oder Zahlen in formatierter Form auf dem Drucker (LPRINT USING) oder dem Bildschirm (PRINT USING).
Beispiele: PRINT USING "###.##";a,b
LPRINT USING "!";txt$

PRINT #<dateinnr>,[USING,<formatkette>;]<liste der ausdrücke>
Ausgabe von Daten in eine sequentielle Diskettendatei in formatierter bzw. unformatierter Form.
Beispiele: PRINT #4,USING "###";c,d
PRINT #3, a$,b

Formatzeichen

Zeichen	Bedeutung	Beispiel
!	Ausgabe des 1. Zeichens der Zeichenkette	"!"
\nLeerzeichen\	Ausgabe von 2+n Zeichen der ZK	"\ \"
&	Zeichenkette wird ohne Veränderung ausgegeben (variable Länge)	
#	definiert eine Ziffernposition	"###"
.	definiert den Dezimalpunkt	"##.##"
+	Ein "+" am Anfang (oder Ende) der <formatkette> bewirkt, daß das Vorzeichen (+|-) vor (oder nach) der Zahl ausgegeben wird.	"+#.##" "##."##+"
-	Ein "-" am Ende der <formatkette> bewirkt, daß das Vorzeichen (-) am Ende ausgegeben wird.	"###-"
**	definiert die Ausgabe von Sternen an Stelle von führenden Leerzeichen.	"**###.#"
$$	definiert die Ausgabe eines "$" unmittelbar vor der Zahl.	"$$###.##"
**$	definiert die Ausgabe von Sternen an Stelle von führenden Leerzeichen und eines "$" unmittelbar vor der Zahl.	"**$##.##"
,	definiert die Tausender-Abtrennung durch Komma	"#####,.##"
^^^^	definiert die Ausgabe der Zahl in der Exponentialform	"##.##^^^^"
_	Ein Unterstreichstrich "_" in der <formatkette> bewirkt, daß die folgenden Textzeichen unverändert ausgegeben werden.	"##_/##"

PUT [#]<dateinr>[,<satznr>]
Ausgabe eines Zeichens aus dem Puffer in eine Direktzugriffsdatei. Ohne Angabe der Satznummer wird immer auf den folgenden Satz zugegriffen.
Beispiele: PUT #4,snr
PUT #4

READ <liste von variablen>
Lesen von Konstanten aus einer DATA-Anweisung und Zuweisung an Variable.
Beispiel: READ i,a%,t$

RSET <zeichenkettenvariable> = <zeichenkettenausdruck>
Rechtsbündiges Eintragen der Daten (<zeichenkettenausdruck>) in den Pufferspeicher (<zeichenkettenvariable>) für Direktzugriffsdateien.
Beispiel: RSET b$="ANFANG"
RSET a$=MKS$(xy$)

WRITE [<liste von ausdrücken>]
Ausgabe von Daten über den Bildschirm.
Beispiel: WRITE A,B,C$

WRITE #<dateinr>,<liste der ausdrücke>
Ausgabe von Daten in eine sequentielle Datei.
Beispiel: WRITE #5,A$,B$

9. Anweisungen / Funktionen für Prozessorarbeit

CALL <name der variablen> [(<parameterliste>)]
Ruf eines in Assemblersprache geschriebenen Unterprogramms (Variable enthält UP-Adresse).
Beispiel: oput=&H9A00
CALL oput(a,b)

DEF USR[<ziffer>] = <integerausdruck>
Spezifizieren der Startadresse einer USR-Funktion in Assemblersprache.
<ziffer> = Nummer der USR-Routine (0,1,...,9)
Beispiel: DEF USR3 = %H9B00

INP (I)
Die Funktion liefert das Byte, welches vom Port I gelesen wurde.
Beispiel: PRINT INP(150)

OUT I,J
Ausgabe eines Bytes (J) über Port (I).
Beispiel: OUT 17,52

PEEK (I)
Die Funktion PEEK liefert das Byte, welches von der Speicheradresse I gelesen wurde.
Beispiel: a% = PEEK (%H3C4A)

POKE I,J
Ausgabe eines Zeichens J an die Speicherstelle I.
Beispiel: POKE &H9841,&H50

USR[<ziffer>] (X)
Ruf eines Assembler-Unterprogramms mit Argument X. Die UP-Adresse ist mit DEF USR zu definieren.
Fehlt die <ziffer>, dann wird USR0 angenommen.
Beispiel: y = USR3 (x+6)

VARPTR (<variablenname>)
Die Funktion liefert die Adresse des 1. Bytes der Daten, die durch <variablenname> gekennzeichnet sind.
Beispiel: a: = VARPTR (x)

VARPTR <#<dateinr>)
liefert die Adresse des Disketten-Ein-/Ausgabepuffers.
Beispiel: j = VARPTR (#2)

WAIT <portnummer>,I[,J]
Unterbrechung der Programmdurchführung und Überwachung des Status eines Eingabeports. Die vom Port gelesenen Daten werden mit J konjunktiv und mit I disjunktiv verknüpft.
Wenn das Ergebnis ungleich Null ist, wird das Programm fortgesetzt.
Beispiel: WAIT 150,3,2

10. Funktionen

Arithmetische Funktionen

ABS(X)
liefert den Absolutwert von X.
Beispiel: y = ABS(a+b)

ATN(X)
liefert den Arcustangens von X.
Beispiel: y = ATN(a)

CDBL(X)
konvertiert X in eine Zahl doppelter Genauigkeit.
Beispiel: y = CDBL(a!)

CINT(X)
konvertiert X in eine Integerzahl.
Beispiel: y = CINT(b)

COS(X)
liefert den Cosinus von X.
Beispiel: y = COS(0.42)

CSNG(X)
konvertiert X in eine Zahl einfacher Genauigkeit.
Beispiel: y = CSNG(1#)

EXP(X)
liefert den Funktionswert e^x.
Beispiel: y = EXP(c)

FIX(X)
liefert den abgetrennten Integerteil von X.
Beispiel: y = FIX(a/b)

FRE(X)
liefert die Anzahl der nicht belegten Bytes (freier Speicher).
Beispiel: y = FRE(1)

INT(X)
liefert die größte Integerzahl <= X.
Beispiel: y = INT(x+3)

LOG(X)
liefert den natürlichen Logarithmus von X.
Beispiel: y = LOG((y-4)

RND(X)
liefert eine Zufallszahl zwischen 0 und 1.
Beispiel: y = RND(4)

SGN(X)
liefert bei X > 0 = 1
           X = 0 = 0
           X < 0 = -1
Beispiel: y = SGN(g)

SIN(X)
liefert den Sinus von X.
Beispiel: y = SIN(a)

SQR(X)
liefert die Quadratwurzel von X.
Beispiel: y = SQR(3*c)

TAN(X)
liefert den Tangens von X.
Beispiel: y = TAN(x/2)

Zeichenkettenfunktionen

ASC(Z$)
liefert einen numerischen Wert, der dem ASCII-Code des 1. Zeichens von Z$ entspricht.
Beispiel: PRINT ASC(a$)

CHR$(X)
liefert das Zeichen, das zu X, gemäß der ASCII-Tabelle, gehört.
Beispiel: PRINT CHR$(48)

FRE("")
Reorganisation (Verdichten) des Speichers.
Beispiel: PRINT FRE ("")

HEX$(X)
liefert eine Zeichenkette, die den hexadezimalen Wert von X darstellt.
Beispiel: z$ = HEX$(100)

INPUT$(X)
Eingabe einer Zeichenkette von X Zeichen.
Beispiel: z$ = INPUT$(8)

INKEY$
Eingabe von einem Zeichen über die Tastatur.
Beispiel: z$ = INKEY$

INSTR ([I,]Z$,Y$)
liefert die Position des 1. Auftretens der Zeichenkette in Z$, beginnend mit dem 1. Zeichen von Z$.
Beispiele: i = INSTR (a$,b$)
i = INSTR (3,a$,b$)

LEFT$ (Z$,I)
liefert die I ersten Zeichen der Zeichenkette Z$.
Beispiel: LEFT$ (z$,i)

LEN (Z$)
liefert die Länge der Zeichenkette Z$.
Beispiel: c = LEN(a$)

MID$ (Z$,I[,J])
liefert eine Zeichenkette der Länge J, beginnend mit dem I. Zeichen von Z$.
Beispiel: a$ = MID$ (b$,5,8)

OCT$ (X)
liefert eine Zeichenkette, die den oktalen Wert von X darstellt.
Beispiel: PRINT OCT$(105)

RIGHT$ (Z$,I)
liefert die rechten I Zeichen von Z$.
Beispiel: a$ = RIGHT$(b$,3)

SPACE$ (X)
liefert eine Zeichenkette von X Zeichen.
Beispiel: PRINT SPACE$(10)

STR$ (X)
konvertiert einen Zahlenwert X in eine Zeichenkette.
Beispiel: a$ = STR$(123)

STRING$ (I,J)
liefert eine Zeichenkette der Länge I, deren Zeichen den ASCII-Code J haben.
Beispiel: a$ = STRING$(8,60)

STRING$ (I,Z$)
liefert eine Zeichenkette der Länge I, in der alle Zeichen dem 1. Zeichen von Z$ entsprechen.
Beispiele: a$ = STRING$(5,"*")
a$ = STRING$(8,zk$)

VAL (Z$)
konvertiert die Zeichenkette Z$ in eine n numerischen Wert.
Beispiele: PRINT VAL("1234")
PRINT VAL(zk$)

Ein-/Ausgabefunktionen

CVI (Z$)
konvertiert die Zeichenkette Z$ in INTEGER-Werte.
Beispiel: y! = CVI(z$)

CVS (Z$)
konvertiert die Zeichenkette Z$ in Gleitkommawerte einfacher Genauigkeit.
Beispiel: a% = CVS(b$)

CVD (Z$)
konvertiert die Zeichenkette Z$ in Gleitkommawerte doppelter Genauigkeit.
Beispiel: d# = CVD(c$)

EOF (dateinr)
gibt "wahr"(-1) aus, wenn das Ende einer sequentiellen oder direkten Datei erreicht ist:
Beispiel: IF EOF(3) THEN END

ERL
liefert die Zeilennummer, in welcher ein Fehler aufgetreten ist.
Beispiel: PRINT "Fehler in Zeile: ", ERL

ERR
liefert die Fehlercode-Nummer.
Beispiel: IFF ERR=61 THEN PRINT "Diskette voll !"

LOC (<dateinr>)
liefert die nächste Satznummer (bei direkten Dateien) bzw. die Anzahl der geschriebenen/gelesenen Sektoren bei sequentiellen Dateien.
Beispiel: IF LOC(4) > 100 THEN END

LPOS (X)
liefert die aktuelle Position des Druckers.
Beispiel: IF LPOS(1) > 60 THEN LPRINT

MKI$ (X%)
konvertiert INTEGER-Werte in Zeichenketten.
Beispiel: LSET z$=MKI$(i%)

MKS$ (Y!)
konvertiert Gleitkommazahlen einfacher Genauigkeit in Zeichenketten.
Beispiel: LSET t$=MKS$(e!)

MKD$ (W#)
konvertiert Gleitkommazahlen doppelter Genauigkeit in Zeichenketten.
Beispiel: LSET tx$=MKD$(d#)

POS (X)
liefert die aktuelle Kursorposition der Konsole.
Beispiel: IF POS(1) > 63 THEN PRINT

SPC (X)
Ausgabe von X Leerzeichen.
Beispiel: PRINT SPC(10)

TAB (X)
Verschiebung Druckkopf/Kursor auf Position X.
Beispiel: LPRINT TAB(80),a$
PRINT TAB(10),"AB"

11. Tabelle Fehler-Codes

1	NEXT ohne FOR	(NEXT without FOR)
2	Syntax-Fehler	(Syntax error)
3	Rücksprung ohne GOSUB	(RETURN without GOSUB)
4	ungültige Daten	(OUT of data)
5	unzulässiger Funktionsaufruf	(illegal function call)
6	Überlauf Daten	(Overflow)
7	Speicherüberlauf	(Out of memory)
8	Zeile nicht definiert	(Undefined line)
9	Index außer Bereich	(Subscript out of range)
10	Redimensionierung Feld	(Redimensioned array)
11	Division durch Null	(Division by zero)
12	unzul. dir. Verarbeitung	(Illegal direct)
13	Typunverträglichkeit	(Type mismatch)
14	außerhalb Zeichenketten-Raum	(Out of string space)
15	Zeichenkette zu lang	(String too lang)
16	Zeichenk.-Formel zu komplex	(String formula too complex)
17	Fortsetzung nicht möglich	(Can't continue)
18	undefinierte Anwender-Funkt.	(Undefined user function)
19	kein RESUME	(No RESUME)
20	RESUME ohne Fehler	(RESUME without error)
21	undefinierter Fehler	(Unprintable error)
22	fehlender Operand	(Missing operand)
23	Zeilenpuffer-Überlauf	(Line buffer overflow)
26	FOR ohne NEXT	(FOR without NEXT)
29	WHILE ohne WEND	(WHILE without WEND)
30	WEND ohne WHILE	(WEND without WHILE)

Diskettenfehler

50 Feldüberlauf	(Field overflow)
51 interner Fehler	(Internal error)
52 falsche Dateinummer	(Bad file number)
53 Datei nicht gefunden	(File not found)
54 falscher Dateimodus	(Bad file mode)
55 Datei bereits eröffnet	(File already open)
57 Disketten E/A-Fehler	(Disk I/O error)
58 Datei existiert bereits	(File already exist)
61 Diskette voll	(Disk full)
62 Eingabe nach log. Ende	(Input past end)
63 falsche Satznummer	(Bad record number)
64 falscher Dateiname	(Bad file name)
66 Direktanweisung in Datei	(Direct statement in file)
67 zu viele Dateien	(Too many files)

12. BASIC-Compiler

12.1. Unterschiede BASIC-Compiler/-Interpreter

Vom Compiler nicht unterstützte Interpreterkommandos:

AUTO	CLEAR	COMMON	ERASE
EDIT	CONT	MERGE	NEW
RENUM	LIST	DELETE	
LOAD	LLIST	SAVE	

12.2 Sprachunterschiede zum Interpreter

CALL <variablenname> [<argumentliste>]
Der <variablenname> ist der Name der Subroutine, die aufgerufen werden soll (muß durch den Programmverbinder als ein globales Symbol erkannt werden).

CHAIN Die Optionen ALL, MERGE, DELETE und <zeilennr> werden vom Compiler nicht unterstützt.

CLEAR Für <ausdruck1> und <ausdruck2> sind nur ganzzahlige Ausdrücke zugelassen.

COMMON Die COMMON-Anweisung muß im Programm vor einer ausführbaren Anweisung stehen. Felder müssen vorher in einer DIM-Anweisung definiert sein.

DEFINT | DEFSNG | DEFDBL | DEFSTR
Diese Anweisungen werden sofort ausgeführt, wenn sie erkannt werden. Ein Überspringen mit GOTO ist nicht möglich.

DIM Die Anweisung wirkt erst dann, wenn sie vom Compiler erkannt wird und kann nicht mehrfach ausgeführt werden. Die Werte der Indizes müssen ganzzahlige Konstanten sein.

END schließt alle Dateien und veranlaßt Rückkehr zum SCP.

FOR/NEXT
Schleifen müssen statisch geschachtelt sein. Laufvariablen doppelter Genauigkeit werden verarbeitet.

FRE mit einem numerischen Parameter bewirkt immer die Ausgabe des Wertes Null.

ON ERROR GOTO/RESUME
Die ON ERROR GOTO - Anweisung mit RESUME <zeilennr> setzt den Compilerschalter /E, mit RESUME, RESUME 0 oder RESUME NEXT den Compilerschalter /X voraus.

RUN Die Option "R" wird nicht unterstützt.

STOP unterbricht den Programmablauf und bewirkt eine Rückkehr zu SCP.

TRON/TROFF
Bei Verwendung dieser Anweisung muß der Compilerschalter /D gesetzt werden.

USER Es erfolgt keine Übergabe von Parametern.

WHILE/WEND
Schleifen müssen statisch geschaltet sein.

%INCLUDE <dateiname>
Einfügen externer Quelldateien (BASIC-Quelle während der Compilierung.
%INCLUDE PR.BAS

12.3 Compilierung von Programmen

Kommandos
BASC [<objektdateiname>][,[<listdateiname>]]=<quelldateiname> [<schalter>...]

Compilerschalter:
/C Quelldatei darf nichtnumerierte Zeilen enthalten.
/E Programm enthält ON ERROR GOTO mit RESUME <zeilennr>.
/X Programm enthält ON ERROR GOTO mit RESUME, RESUME 0 oder RESUME NEXT.
/Z Bei der Übersetzung wird der Z80-Operationscode verwendet.
/N Unterdrückung des Auflistens vom reassemblierten Objektcode in der Listdatei.
/D Generierung Testcode für Laufzeitfehlerprüfung mit TRON/TROFF.
/S Zeichenkettenkonstanten > 4 Zeichen werden in REL-Datei (nicht in Speicher) geschrieben.

Beispiele:
(1) BASC TEST,PROT=TEST/Z

Das Quellprogramm TEST.BAS wird übersetzt in die Objektdatei TEST.REL. Gleichzeitig wird eine Listdatei PROT.PRN ausgegeben.

(2) BASC TEST=TEST/Z
Analog (1); eine Listdatei wird unterdrückt.

(3) BASC TEST,TTY:=B:TEST/Z/X/D
Das Quellprogramm TEST.BAS vom Laufwerk B wird in TEST.REL (auf aktuellem Laufwerk) übersetzt. Die Listdatei wird auf dem Bildschirm ausgegeben.

(4) BASC TEST,LST:=B:TEST/Z/S
Analog (3); die Listdatei wird auf dem Drucker ausgegeben.

(5) BASC ,=TEST
Das Quellprogramm TEST.BAS wird syntaktisch geprüft. Eine Objektdatei wird nicht ausgegeben.

(TTY: = Ausgabe LIST-Datei auf den Bildschirm;
LST: = Ausgabe LISTdatei auf den Drucker)

12.4 Verbinden von Programmen

Kommandos:
LINK <objektdateiname 1>[,<objektdateiname 2>...] <schalter>

Schalter für Programmverbinder:
/E Eine Zieldatei mit der Klassenbezeichnung .COM wird auf die Diskette übertragen. Die notwendigen Routinen werden aus BASLIB.REL gelesen und an das Zielprogramm gebunden. Nach Übersetzung erfolgt ein Rücksprung zum SCP.

/N Der Schalter /N hinter einem Dateinamen informiert den Binder, daß dieser Name beim Schreiben der COM-Datei zu verwenden ist.

/G Der Schalter /G entspricht dem Schalter /E, jedoch wird das erzeugte Zielprogramm sofort ausgeführt.

Beispiele:
(1) LINK TEST, UP, PROG/N/E
Die Programme TEST und UP werden geladen, mit den notwendigen Routinen aus BASLIB.REL gebunden und unter dem Namen PROG.COM auf Diskette ausgegeben. Anschließend erfolgt ein Rücksprung zum SCP.

(2) LINK PRG/N, PRGQ/G
Das Programm PRGQ wird geladen, mit den notwendigen Routinen von BASLIB.REL verbunden und als PRG.COM auf Diskette ausgegeben. Anschließend wird das Programm gestartet.

12.5 Fehlermeldungen vom Compiler

Fehler (Fatal Errors):

SN	Syntax-Fehler	(Syntax error)
OM	Speicherüberlauf	(Out of memory)
SQ	Sequenzfehler	(Sequence error)
TM	Typenunverträglichkeit	(Type mismatch)
TC	zu hohe Komplexität	(Too complex)
BS	unzulässige Indizierung	(Bad subscript)
LL	Zeile zu lang	(Line too long)
UC	nicht definiertes Kommando	(Unrecognizable command)
OV	Zahlenüberlauf	(Math overflow)
IO	Division durch Null	(Division by zero)
DD	Feld bereits dimensioniert	(Array allready dimensioned)
FN	FOR/NEXT-Fehler	(FOR/NEXT error)
FD	Funktion bereits definiert	(Function already defined)
UF	Funktion nicht definiert	(Function not defined)
WE	WHILW/WEND-Fehler	(WHILE/WEND rror)
/E	fehlender /E-Schalter	(Missing /E switch)
/X	fehlender /X-Schalter	(Missing /X switch)

Warnungen (Warning Errors)

ND	Feld nicht dimensioniert	(Array not dimensioned)
SI	Anweisung ignoriert	(Statement ignored)

12.6 Fehlernachrichten vom Laufzeitsystem

2	Syntax-Fehler	(Syntax error)
3	RETURN ohne GOSUB	(RETURN without GOSUB)
4	ungültige Daten	(Out of data)
5	unzulässiger Funktionsruf	(Illegal function call)
6	Gleitkomma-/Integerüberlauf	(Floating overflow or integer overflow)
9	Index außerhalb des Werte-bereiches	(Subscript out of range)
11	Division durch Null	(Division by zero)
14	außerhalb Zeichenkettenraum	(Out of string space)
20	RESUME ohne Fehler	(RESUME without error)
21	nichtdefinierter Fehler	(Unprintable error)
50	Feldüberlauf	(Field overflow)
51	interner Fehler	(Internal error)
52	falsche Dateinummer	(Bad file number)
53	Datei nicht gefunden	(File not found)
54	falscher Dateimodus	(Bad file mode)
55	Datei bereits geöffnet	(File already open)
57	Disketten E/A-Fehler	(Disk I/O error)
58	Datei existiert bereits	(File already exists)
61	Diskette voll belegt	(Disk full)
62	Eingabe nach logischem Ende	(Input past end)
63	falsche Satznummer	(Bad record number)
64	unzulässiger Dateiname	(Bad file name)
67	zu viele Dateien	(Too many files)

Druck: VEB Dienstleistungsbetrieb Erfurt V/4/56 R1 01/86

robotron

VEB Robotron Büromaschinenwerk
»Ernst Thälmann« Sömmerda
Weißenseer Straße 52
Sömmerda
DDR – 5230

Exporteur:
Robotron Export–Import
Volkseigener
Außenhandelsbetrieb
der Deutschen
Demokratischen Republik
Allee der Kosmonauten 24
Berlin
DDR – 1140